Standard Code Library

FLself

SCUT

August 15, 2022

Contents

一切的开始

一些宏定义

- 需要 C++11

```cpp
// #define DEBUG
// #define InTerminal
// #define std_cpp17
#include<bits/stdc++.h>
#define int long long
#define PII std::pair<int, int>
#define VI std::vector<int>
#define VPII std::vector<std::pair<int, int> >
#define VVI std::vector<std::vector<int> >
#define ALL(a) (a).begin(), (a).end()
#define SIZ(a) ((int)(a).size())
#define FOR(i, l, r) for (int i = (l); i <= (r); ++i)
#define REP(i, r, l) for (int i = (r); i >= (l); --i)
#define lowbit(x) ((x) & (-(x)))
#define lbpos(x) (__builtin_ctz(x))
#define hbpos(x) (31 - __builtin_clz(x))

template<typename S, typename T> std::istream &operator>>(std::istream &is, std::pair<S, T> &pp) { is >> pp.first >> pp.second; return is; }
template<typename S, typename T> std::ostream &operator<<(std::ostream &os, std::pair<S, T> pp) { os << "(" << pp.first << ", " << pp.second << ")"; return os; }
template<typename S, std::size_t _siz> std::istream &operator>>(std::istream &is, std::array<S, _siz> &arr) { for (auto &x: arr) is >> x; return is; }
template<typename S, std::size_t _siz> std::ostream &operator<<(std::ostream &os, std::array<S, _siz> arr) { os << "("; for (auto x: arr) os << x << ", "; os << ")"; return os; }
template<typename T> std::istream &operator>>(std::istream &is, std::vector<T> &vec) { for (auto &x: vec) is >> x; return is;}
template<typename T> std::ostream &operator<<(std::ostream &os, const std::vector<T> &vec) { os << '{'; for (auto &x: vec) os << x << ", "; return os << "}";}
#ifdef std_cpp17
template<class Tuple, std::size_t... Is> void print_tuple_impl(std::ostream &os, const Tuple &t, std::index_sequence<Is...>) { ((os << (Is == 0? "" : ", ") << std::get<Is>(t)), ...); }
template<class... Args> std::ostream &operator<<(std::ostream &os, const std::tuple<Args...> &t) { os << "("; print_tuple_impl(os, t, std::index_sequence_for<Args...>{}); return os << ")"; }
#endif
#ifdef DEBUG
#ifdef InTerminal
#define dbg(x...) do { std::cerr << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { std::cerr << "\033[39;0m" << std::endl; }
#else
#define dbg(x...) do { std::cerr  << #x << " -> "; err(x); } while (0)
void err() { std::cerr << std::endl; }
#endif
template<typename T, typename... A>
void err(T a, A... x) { std::cerr << a << ' '; err(x...); }
#else
#define dbg(...)
#endif

using namespace std;
const int maxn = 2e5 + 3;
const int INF = 0x3f3f3f3f3f3f3f3f;
const int mod = 998244353;
mt19937 RD(time(0));



void solv() {


    return ;
}

signed main() {
    // freopen("./data.in", "r", stdin);
```

```
    std::ios::sync_with_stdio(false), std::cin.tie(0), std::cout.tie(0);
    int beg__TT = clock();

    signed _ttt;
    cin >> _ttt;

    while(_ttt--)
        solv();

    #ifdef DEBUG
    std::cerr << "use : " << (clock() - beg__TT) << "ms\n";
    #endif
    return 0;
}
```

数据结构

ST 表

- 一维

```
class Sparcetable {
    vector<vector<int> > st;
    int siz;
    bool MX_flg = 0;
    inline int renew(int x, int y) {
        if (MX_flg) return  max(x, y);
        return min(x, y);
    }
public:
    // 注意 bhpos(0) 返回-1
    bool (*comp)(int, int);
    Sparcetable():siz(maxn) {st.resize(hbpos(maxn - 1) + 1, std::vector<int> (maxn));}
    Sparcetable(const std::vector<int>& a, bool _MX_flg = 1): siz(a.size()), MX_flg(_MX_flg) {
        int n = a.size();
        st.resize(hbpos(n) + 1, vector<int> (n + 1));
        for (int i = 1; i <= n; ++i) st[0][i] = a[i];
        for (int i = 1; i <= hbpos(siz); ++i) {
            for (int j = 1; j + (1 << i) <= siz + 1; ++j) {
                st[i][j] = renew(st[i - 1][j], st[i - 1][j + (1 << (i - 1))]);
            }
        }
    }
    int query(int l, int r) {
        int len = hbpos(r - l + 1);
        return renew(st[len][l], st[len][r - (1 << len) + 1]);
    }
};
```

- 二维

```
int f[10][10][maxn][maxn];
#define _highbit(x) (31 - __builtin_clz(x))
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[p][q][x][y], f[p][q][xx - (1 << p) + 1][yy - (1 << q) + 1]),
        max(f[p][q][xx - (1 << p) + 1][y], f[p][q][x][yy - (1 << q) + 1])
    );
}
void init() {
    for (int x = 0; x <= _highbit(n); ++x)
    for (int y = 0; y <= _highbit(m); ++y)
        for (int i = 0; i <= n - (1 << x); ++i)
        for (int j = 0; j <= m - (1 << y); ++j) {
            if (!x && !y) { f[x][y][i][j] = a[i][j]; continue; }
            f[x][y][i][j] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
```

```
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, _highbit(xx - x + 1), _highbit(yy - y + 1));
}
```

Fenwick Tree(树状数组)

- 一维

```
template<typename T>
class FenwickT {
    int n;
    vector<T> tr;
public:
    FenwickT(int siz): tr(siz), n(siz) {}
    FenwickT(int siz, T ini): tr(siz, ini), n(siz) {}
    void add(int p, T x) {
        for (int i = p; i < n; i += i & (-i)) tr[i] += x;
    }
    T query(int p) {
        T ret = T(0);
        for (int i = p; i > 0; i -= i & (-i)) ret += tr[i];
        return ret;
    }
    T range_sum(int l, int r) {
        return (query(r) - query(l - 1));
    }
};
```

数学

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i\lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时, $h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$; 否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
```

```
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// ------------------------------------------------
```

字符串

KMP

可以尝试从 $fail$ 树的角度理解

```
bool kmp(string &s, string &p) {
    int n = p.size();
    vector<int> nex(n + 1);
    for (int i = 1, l = 0; i < n; ++i) {
        while (l && p[l] != p[i]) {l = nex[l];}
        if (p[l] != p[i]) nex[i] = 0;
        else nex[i] = ++l;
    }
    n = s.size();
    for (int i = 0, now = 0; i < n; ++i) {
        while (now && s[i] != p[now]) now = nex[now - 1];
        if (s[i] == p[now]) ++now;
```

```
        if (now == p.size()) return true;
    }
    return false;
}
```

后缀自动机

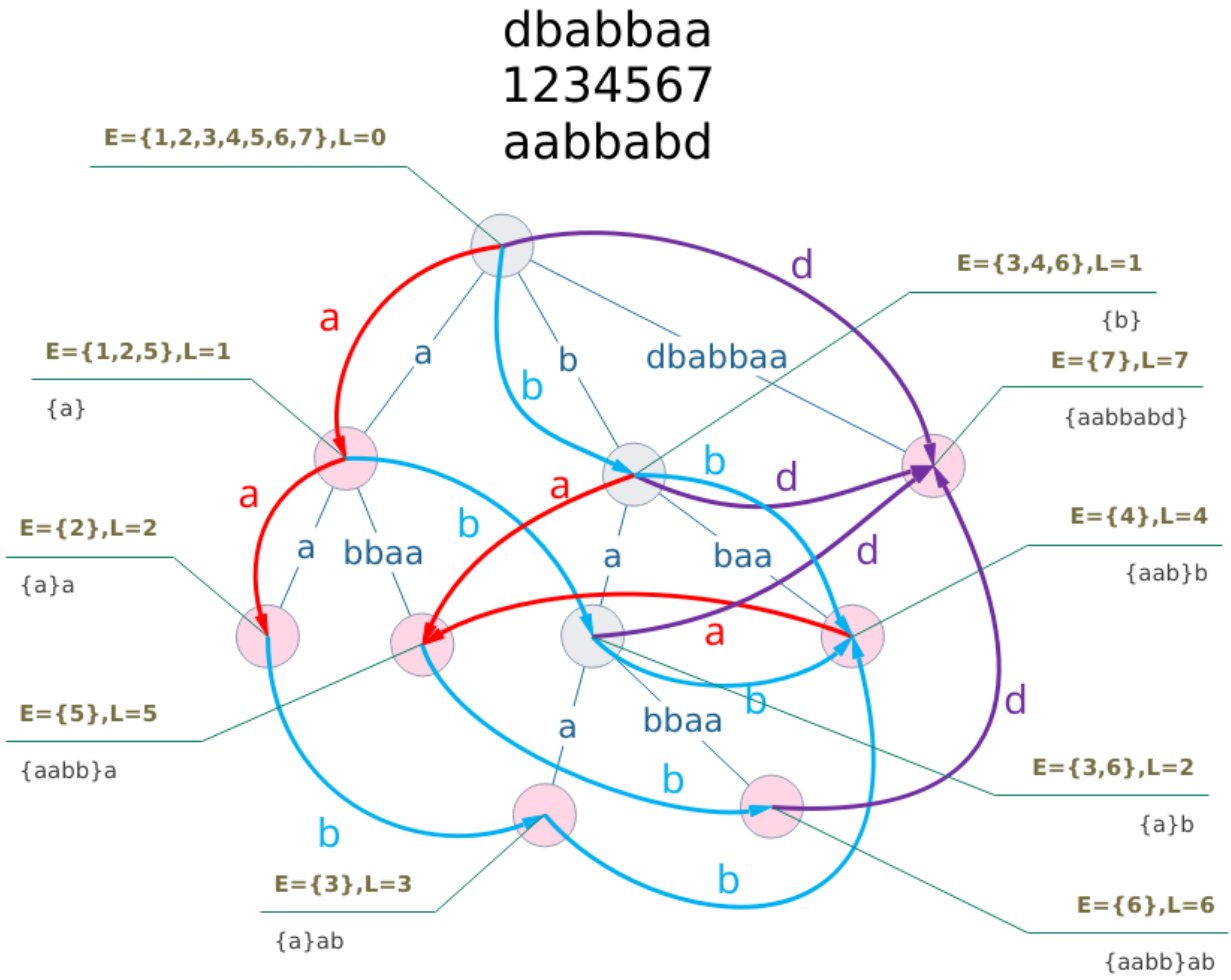

Figure 1: 后缀自动机图解

```
struct SuffixAutomaton {
    static constexpr int ALPHABET_SIZE = 26;
    int N = 1e5;
    struct Node {
        int len;
        int link;
        // int next[ALPHABET_SIZE];
        std::vector<int> next;
        Node() : len(0), link(0), next(ALPHABET_SIZE) {}
    };
    std::vector<Node> t;
    int cntNodes;
    int extend(int p, int c) {
        if (t[p].next[c]) {
            int q = t[p].next[c];
            if (t[q].len == t[p].len + 1)
                return q;
            int r = ++cntNodes;
            t[r].len = t[p].len + 1;
```

```
            t[r].link = t[q].link;
            // std::copy(t[q].next, t[q].next + ALPHABET_SIZE, t[r].next);
            t[r].next = t[q].next;
            t[q].link = r;
            while (t[p].next[c] == q) {
                t[p].next[c] = r;
                p = t[p].link;
            }
            return r;
        }
        int cur = ++cntNodes;
        t[cur].len = t[p].len + 1;
        while (!t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
        }
        t[cur].link = extend(p, c);
        return cur;
    }
    SuffixAutomaton(int N_): t(2 * N_), N(N_) {
        cntNodes = 1;
        // std::fill(t[0].next, t[0].next + ALPHABET_SIZE, 1);
        t[0].next.assign(ALPHABET_SIZE, 1);
        t[0].len = -1;
    }
    SuffixAutomaton(string s): t(2 * s.size()), N(s.size()) {
        cntNodes = 1;
        t[0].next.assign(ALPHABET_SIZE, 1);
        t[0].len = -1;

        int p = 1;
        for (auto ch: s) {
            p = extend(p, ch-'a');
        }
    }
};
```

杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```